Music Genre Classification using Machine Learning

Introduction to Machine Learning (CS412) - Group Project

Monisha Siddananda Sampath, Neha Damele, Praveen Raj Veluswami

Computer Science Department
University of Illinois at Chicago
Chicago, Illinois

Content

1. Abstract

Music is one of the most researched topics in applications of machine learning and artificial intelligence. It has proven its worth by attracting several technology companies like Spotify, Amazon, Google, Warner Bros., etc. to invest in it significantly. Differentiating and finding similarities between songs is an area where many of them are working on. This is an area of interest for both music experts and companies developing commercial products. Music or audio processing in general is an interdisciplinary study. In this project, we aim to classify given audio into one of ten different genres. We explore different models and techniques to arrive at the best possible model to classify music into respective genres.

2. Data and Features

The dataset being used in this project is GTZAN Genre Collection from Marsyas. Marsyas (Music Analysis, Retrieval, and Synthesis for Audio Signals) is an open-source software framework for audio processing with a specific emphasis on Music Information Retrieval applications.[1]

The dataset contains 10 genres with 100 tracks in each genre i.e. in a total of 1000 tracks. Each track is 30 seconds long. The tracks are all 22050 Hz Mono 16-bit audio files in .wav format.
The dataset was further processed by dividing each track by 10 i.e. 3 seconds long and 57 features were extracted from each track thus providing a dataset of 10000 rows and 57 features.[2]

3. Approach / Method

3.1 Exploratory Data Analysis

We do exploratory data analysis (EDA) on our dataset to gain insights into the data we are using, which might prove to be useful to improve the performance of our models. We perform the following EDA on our dataset.

Descriptive Statistics: We get the statistical information from our dataset. Knowing the 75th percentile, maximum values, etc. will help us to determine whether a given feature has any outliers. Shown in figure 1 are the descriptive statistics of the first four features in the dataset.

	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var
count	9990.000000	9990.000000	9990.000000	9.990000e+03
mean	0.379534	0.084876	0.130859	2.676388e-03
std	0.090466	0.009637	0.068545	3.585628e-03
min	0.107108	0.015345	0.000953	4.379535e-08
25%	0.315698	0.079833	0.083782	6.145900e-04
50%	0.384741	0.085108	0.121253	1.491318e-03
75%	0.442443	0.091092	0.176328	3.130862e-03
max	0.749481	0.120964	0.442567	3.261522e-02

Figure 1: Table showing the descriptive statistics of some of the features in our dataset

Correlation methods: We compute the correlation coefficients between two features to find the relationship between them both. For example, the correlation coefficient computed between the features *mfcc1_mean* and *mfcc2_mean* was found to be -0.6.

Null value/Invalid value inspection: Null values or invalid values found in a dataset can sometimes heavily influence the performance of a dataset. Hence, it is essential to handle null/invalid values in the dataset, if present. However, we did not find any null/invalid values in the dataset.

3.2. Preprocessing data

Data preprocessing is an important step in any machine learning pipeline. Datasets obtained from the real world are often imperfect. These datasets, if used as it is may dampen the performance of our model. We handle this in the following ways.

Removal of duplicate rows: There were 143 rows that were identified to have identical values to other rows in the dataset. These were removed resulting in 9847 rows in the dataset.

Data standardization: It is important to have features across the dataset to have their respective values standardized. Values in each feature in the dataset were standardized.

Outliers: During data collection, isolated factors might contribute to values that are either too high or too low. It is essential to handle these values, since they might skew the distribution of a particular feature and might lead to inaccurate results. Outliers were first visualized for some features as shown below for the feature - chroma_stft_mean. Boxplots were constructed for some of the features using the 'seaborn' library to do so.

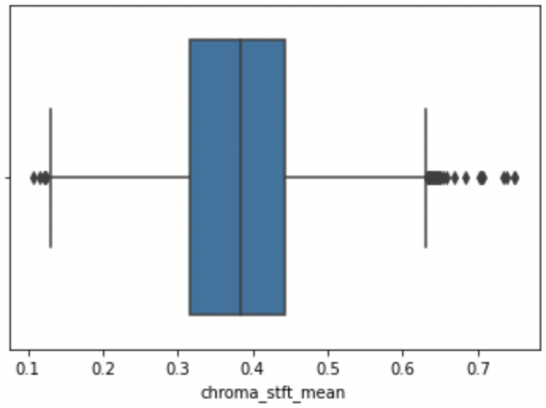

Figure 2: Boxplot for the feature - chroma_stft_mean. As can be seen, there are outliers with values higher than the upper whisker and lower than the lower whisker.

For each feature, the interquartile range was calculated, and thus the lower and upper whiskers. In each feature, values lower than the lower whisker were replaced with lower whisker, and the ones higher than the upper whisker were replaced with the upper whisker.

3.3 Evaluation metrics

The following evaluation metrics were used:

a. Accuracy: It measures how many of the true results are among the total number of predictions.

Accuracy = TP + TN / (TP + TN + FP + FN)

b. Precision: It measures how many of the truly predicted samples are true positives.

Precision = TP / (TP + FP)

c. Recall: It measures how much of the actual true samples are truly classified.

Recall = TP / (TP + FN)

d. F1-score: It is the harmonic mean of precision and recall. The harmonic mean is the mean of the numbers divided by the sum of the reciprocal of the numbers. It is used in cases where we need to measure the average rate. It tends towards the smaller numbers and it is not affected by larger numbers.

F1-score offers a balance between precision and recall. It favors classifiers with similar precision and recall.

F1-score = 2 * (Precision * Recall) / (Precision + Recall)

e. Confusion matrix: It is a N x N matrix, where N is the number of classes. For better visualization of the performance of the classifier the True positive, true negative, false positive, and false negatives are all plotted in a matrix. It helps us to understand which class is correctly classified and which is wrongly classified.

3.4 Feature Selection

Too many features make a machine learning model complex and can reduce its performance. Here we are removing irrelevant features that don't add anything to improve the model's performance. We

have tackled it using Sklearn's Recursive Feature Elimination (RFE), which selects features based on how they affect a particular model's performance. It decreases model complexity by deleting characteristics one by one until only the most important ones remain.

We started with 57 features to train our modal using the XGBoost classifier and got an accuracy of 87%.

Next, we used the RFECV() feature selection tool in Scikit-Learn to identify the features that added no value to the model's performance.

We used features_drop_array = list(np.where(rfecv.support_ == False)[0]) and dropped the following feature from the datalist.

features dropped :

```
['zero_crossing_rate_var', 'mfcc11_var', 'mfcc13_var', 'mfcc14_var',
   'mfcc15_var', 'mfcc16_var', 'mfcc17_var', 'mfcc18_var', 'mfcc19_mean',
   'mfcc20_mean']
```

Even after dropping the above 10 features, we still got an impressive score of 89%.

3.5 Hyperparameter tuning

Hyperparameters are specific variables or weights that govern how an algorithm learns.
Hyperparameter tuning helps in determining the optimal tuned parameters and returns the best fit model.
In our project, we have used a Python module called HYPEROPT, which is based on Bayesian optimization. HYPEROPT searches through a hyperparameter space of values and finds the best possible values that yield the minimum of the loss function.
The XGBoost algorithm uses multiple parameters. To improve the model, parameter tuning is a must.
The hyperopt algorithm on the XGBoost classifier gave the following optimal parameter values:

```
params: {'n_estimators': 1659.0, 'reg_lambda': 92.0}
```

After hypertuning with the best prams given by hyperopt, XGBoost gave 89% accuracy.

Also, tried hypertuning an ensemble classifier, Random Forest.
Got the following optimal parameter values for Random Forest:

```
Best: {'criterion': 1, 'max_depth': 14.0, 'n_estimators': 2}
```

Accuracy improved only slightly from 85% to 86%

3.6 Principal Component Analysis

Principal Component Analysis (PCA) is a technique that helps us reduce the dimension of our dataset. It transforms a large dataset into a smaller one yet retains the variation as much as possible. Our dataset has 57 features in total on which if we perform PCA we might be able to reduce the dimension and hopefully better accuracy on our models too.

Step 1: We first perform PCA on our dataset and evaluate Random Forest and XGB on it. We get 82% and 84% accuracy respectively.

Step 2: To get better accuracy, we take similar features like mfcc and spectral features, perform PCA on them alone, then evaluate our models. Now we get the scores of 83% and 85% respectively, which are better than before but still less than what we obtained previously.

Step 3: Every feature in our dataset except *tempo* has a mean and variance column. We removed the variance columns and performed PCA using the above two methods. Despite getting an improved performance of 85% on Random Forest and 86%

on XGB, it was still lower than what was established in the previous section. The detailed scores can be seen in tables 1 and 2.

	precision	recall	f1-score	support
blues	0.90	0.81	0.85	356
classical	0.88	0.97	0.92	337
country	0.68	0.83	0.75	309
disco	0.85	0.81	0.83	335
hiphop	0.84	0.85	0.84	311
jazz	0.86	0.82	0.84	333
metal	0.93	0.93	0.93	326
pop	0.85	0.89	0.87	323
reggae	0.85	0.83	0.84	309
rock	0.86	0.73	0.79	311
accuracy			0.85	3250
macro avg	0.85	0.85	0.85	3250
weighted avg	0.85	0.85	0.85	3250

Table 1: The precision, recall, f1, and support scores for Random Forest in Step 3

	precision	recall	f1-score	support
blues	0.86	0.85	0.86	356
classical	0.91	0.95	0.93	337
country	0.72	0.83	0.77	309
disco	0.85	0.82	0.84	335
hiphop	0.85	0.86	0.85	311
jazz	0.89	0.85	0.87	333
metal	0.95	0.91	0.93	326
pop	0.89	0.89	0.89	323
reggae	0.88	0.81	0.85	309
rock	0.80	0.79	0.80	311
accuracy			0.86	3250
macro avg	0.86	0.86	0.86	3250
weighted avg	0.86	0.86	0.86	3250

Table 2: The precision, recall, f1, and support scores for the XGB in step 3

4. Result

4.1 Confusion matrix

Fig.4.1 shows the results of the models after feature selection and hyperparameter tuning was performed. The accuracy of both the models is improved by 1% for the Random Forest classifier and 2% for XGB Classifier.

	Baseline	After improvement
Random Forest	85	86
XGB Classifier	87	89

Fig.4.1 Accuracy of Random Forest and XGB classifier with baseline and after improvements

4.2 Classification report

Fig.4.2 shows the confusion matrix of the XGB classifier. The classical, metal, jazz, and pop music genres have the highest f1-scores while country, disco, reggae, and rock have the lowest f1-scores. Fig 4.3 shows that label 1(classical) is correctly classified most of the time. It is misclassified mostly as the label 5(jazz). Also label 2 (country) is one of the genres with the least f1-scores, which has the most misclassifications with label 1 (blues) and label 5 (jazz).

	precision	recall	f1-score	support
blues	0.86	0.92	0.89	352
classical	0.94	0.95	0.95	323
country	0.85	0.85	0.85	324
disco	0.89	0.86	0.88	314
hiphop	0.90	0.87	0.89	336
jazz	0.87	0.92	0.89	332
metal	0.94	0.91	0.92	322
pop	0.91	0.93	0.92	327
reggae	0.91	0.86	0.88	308
rock	0.86	0.85	0.85	312
accuracy			0.89	3250
macro avg	0.89	0.89	0.89	3250
weighted avg	0.89	0.89	0.89	3250

Fig.4.2 Classification report for XGB classifier

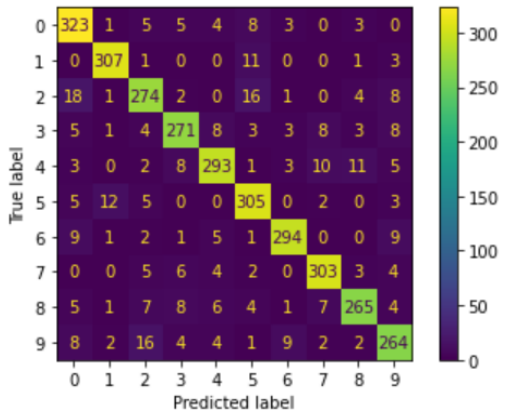

Fig. 4.3 Confusion matrix for XGB classifier

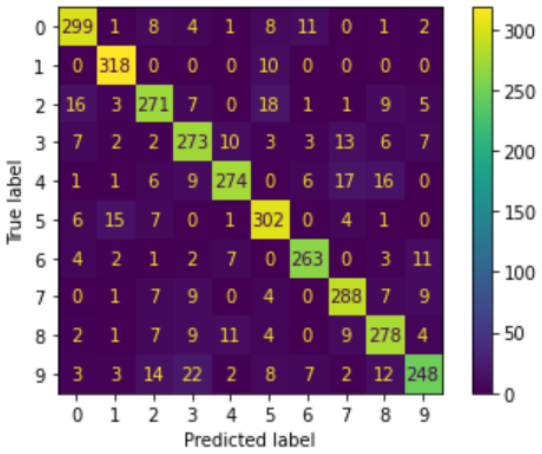

Fig.4.5 Confusion matrix for Random Forest classifier

Fig.4.4 shows the confusion matrix of the Random Forest classifier. Similar to the results of the XGB classifier classical, metal, jazz, and pop music genres have the highest f1-scores while country, disco, and rock have the lowest f1-scores. The f1-score of the rock genre is the lowest with 82. From the confusion matrix in fig.4.5, we can see that the rock genre is misclassified as label 3 (country), label 4 (disco), and label 8 (reggae).

	precision	recall	f1-score	support
blues	0.88	0.89	0.89	335
classical	0.92	0.97	0.94	328
country	0.84	0.82	0.83	331
disco	0.81	0.84	0.83	326
hiphop	0.90	0.83	0.86	330
jazz	0.85	0.90	0.87	336
metal	0.90	0.90	0.90	293
pop	0.86	0.89	0.87	325
reggae	0.83	0.86	0.84	325
rock	0.87	0.77	0.82	321
accuracy			0.87	3250
macro avg	0.87	0.87	0.87	3250
weighted avg	0.87	0.87	0.87	3250

Fig. 4.4 Classification report for Random Forest classifier

4.3 Performance comparison of different classifiers

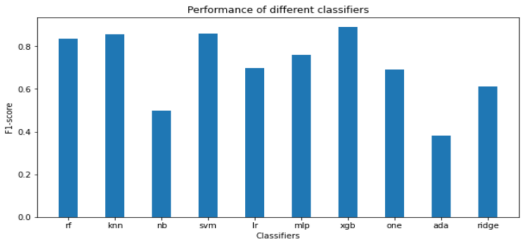

Fig.4.4 F1-Scores of different classifiers

Fid 4.4 shows the f1-score of different classifiers. XGB and Random Forest classifiers have the highest f1-scores.

5. Analysis

The best models were the Random Forest and the XGB Classifier. On analyzing the feature importance in both the classifiers it was found that the perceptr_var is the most important feature as shown in figure 5.1 in figure 5.2. Perceptr features help to understand the shock and rhythm of the song. The second important feature is chroma_stft_mean. Chroma features capture the pitch of the music into twelve bins. They will help

to characterize the harmonic and melody of the music. Stft is the short time fourier transform measured on local sections of the audio file. Spectral bandwidth mean is the third important feature that extracts the spectral bandwidth from the music. It is the extent of the power transform function around the central mean.

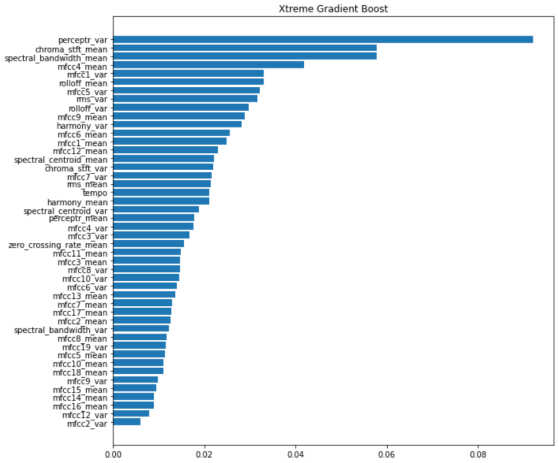

Fig.5.1. Feature importance of XGB classifier

From fig. 4.4 shows that the Naive Bayes classifier performed less when compared to other classifiers. In general, Naive Bayes is a powerful classifier and is generally used in span filters, document classification, etc. But it has a strong assumption that each feature is not dependent on other features, which is not true in this application. Several features combine and contribute to the model which is vital in music genre classification. That may be the reason why Naive Bayes performed less in this application. Ada classifier from sklearn also performed less (fig 4.4). XGB and Ada classifiers are both boosting techniques. The only difference between them is that the Ada classifier changes the misclassified samples by adding weight while gradient boost trains the classifier on the misclassified samples. Also, Ada classifiers are best suited for binary classification. There is a different version of Ada for multiclass classification, but it has been proven that XGBoost is faster than Ada boost.

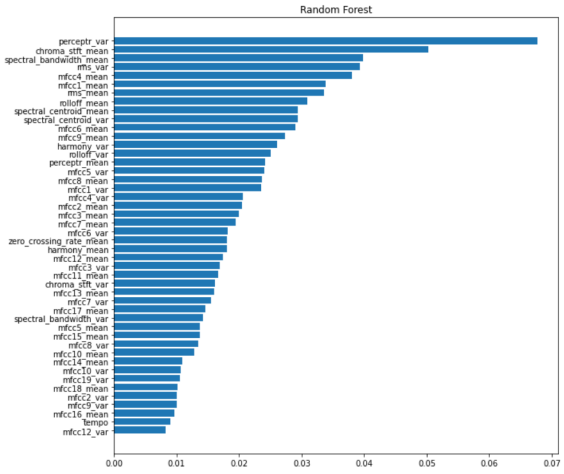

Fig. 5. 2. Feature importance of Random Forest classifier

6. Conclusion

From the baseline classifiers which we chose XGBoost gave the highest accuracy of 89%. XGB gave such high accuracy because of the extreme boosting. It is built on top of regression trees and improved by using gradient boosting. It also has parallel processing which also makes it efficient and portable. The data which we used for this project is sparse with 30 songs in each genre of the length of 30 seconds. The data are extracted from the songs by splitting each song into 3 seconds which gives us 9000 rows total. More data would help the model to understand the data better.

7. Future Scope

- Classification with multiple labels:
  Our project is focused on classifying tracks into a single class, but we can further extend this task by categorizing musical items into multiple labels. For example, one song can belong to Disco and K-Pop simultaneously

- Music Recommendation System:
  Once the model understands what the user likes based on the user's watch history or

similar user's watch history, the recommender system can predict and recommend songs a user might like.

- Song Identifier:
  Can build an app that can identify the music or where people can hum and find the relevant song.

- Novelty detection (not belonging to any class):
  For realistic music classification, the detection of fresh or unknown data that a machine learning system is unaware of during training should be classified as unknown rather than misclassifying.